Blackjack Final Project

My goals were to create an interactive Blackjack game that keep track of the player’s and dealer’s hand, and allowed the player to hit or stay. The game would then ultimately indicate whether the player win, loss, or tied. I created four classes and used two other classes to make the entire game. The additional classes are the Draw class which I used to draw everything on the screen and the Draw Listener class which was used to implement the mouse interactions.

The classes I created were the Card, Deck, Hand, and Game class. First, the Card class was created because the card was the most basic object in the game. Each Card was to store a value and an image.

public Card(String picture, int value)
{
    this.value = value;
    this.picture = picture;
}

Second, the Deck class was used to create an array of 52 cards and then have methods to shuffle the deck and the deal a card from the deck. Then, I initiated the values and images for each cell in the array. For the value of the cards I manually assigned the value to each cell. However for the pictures I wrote a loop to assign the pictures. I had labeled the pictures in order by number so I could easily loop the cards. Also, since the Draw class uses a String to draw a picture, I made sure that the name was being looped into the card as a String.

    for (int i = 0; i < 52; i++)
    {
        int picture_number = i+1;
        String picture = picture_number + "_.gif";
        cards[i] = new Card(picture, values[i]);
    }
The first method in the Deck class is shuffleCards() which was created by exchanging random cards in the array with another cell. Then, it reset the current card to zero so the first card dealt would be the first in the deck. Another important method is dealCard(), which will deal the current card and then change the current card to the next card so the next card dealt is the next card in the array.

```java
public Card dealCard()
{
    Card card = cards[current];
    current++;
    return card;
}
```

Third, the Hand class was used to keep track of the player’s and dealer’s hands. The Hand consists of an array of 10 null cards that will hold a card every time a card is dealt into that hand. So, there is a method that will update the value of the Hand when a card is added to the hand.

```java
public void addCard(Card c)
{
    cards[count] = c;
    value += cards[count].getValue(); //add value of the card
}
```

After a card is added, it will check for the ACE rule. The Ace rule is that Ace can be either a value of 1 or 11 and it will be the value that will make the better hand. The method checks for this rule by checking if the card being added is an ACE (which starts at a value of 11) and if the value of the hand is greater than 21 with the ACE. If so, then the value of the hand will be decreased by, simulating that the value of the Ace will change from 11 to 1. Also, I needed to check if there is already an Ace in the hand and the value is greater than 21, if so then the value of the hand needs to be decreased.

```java
if(c.getValue() == 11 && value > 21)
```
Lastly, the Game class created the Blackjack game. The constructor creates and sets the window for the game. The newGame() method will redraw the table, create a new deck, and new hands for the player and dealer. Then, it shuffled the deck once more and then dealt two cards to the player’s hand and one to the dealer’s hand. After dealing the cards, it checked to see whether the first two cards dealt to the player added to 21 which is a winning blackjack. Also, the win and loss counter is printed.

```java
public void newGame()
{
    ...
    deck = new Deck();
    player_hand = new Hand(); //Create new hands
    dealer_hand = new Hand();
    ...
    deck.shuffleCards();

    //deal and draw player's starting 2 cards
c = deck.dealCard();
    player_hand.addCard(c);
    draw.picture(100, 180, c.getPicture());
    ...
    if(player_hand.getValue() == 21)
    {
        message = "BLACKJACK";
        draw.setPenColor(Color.BLACK);
        draw.text(300, 310, message);
        wins++;
    }
    ...
}
//Win Lose Counter
draw.setPenColor(Color.BLACK);
message = ("Wins: " + wins + " Losses: " + loses);
draw.text(480,35, message);
}

The Hit method deals a card, adds it to the player’s hand, then draw the card on the screen. Once again it will check to make sure the hand value does not exceed 21, if the player’s hand does exceed 21 from too many hits then the “you loose!” message will appear.

public void hit()
{
    c = deck.dealCard(); //deal card
    player_hand.addCard(c); //add card to player's hand
    draw.picture(100 + cn_p, 180, c.getPicture()); //draw card

    //if player's hand is greater than 21 then game over
    if(player_hand.getValue() > 21)
    {
        message = "You loose! (Hand Value:"+player_hand.getValue()+")";
        loses++;
        ...
    }
}

The Stay method will mean the player’s turn is over and the dealer’s hand will be dealt and there will be a win/lose checker. The dealer’s rule is to continue to take cards ("hit") until his total is 16 or greater. I did a while loop to keep adding a card to the hand until the hand is greater than 17. Then it will check for three things, a win, push (tie), loss with if else statements. To check a win, it will check if the value of the player is created than the value of the dealer and that the value of the player’s hand is 21 or less. To check a tie, it just checks that the value of the player and dealer are equal. Lastly, if it the decision isn’t a loss or a tie than it was a win. At the end of each decision the win or loss counter would be updated.

public void stay()
{
...
    while(dealer_hand.getValue() < 17)
    {

Kong 5

c = deck.dealCard(); //deal card
dealer_hand.addCard(c); //add to dealer's hand
draw.picture(100 + cn_d, 450, c.getPicture()); //draw card
cn_d = cn_d + 100; //change player's card placement by 100 pixels

if(dealer_hand.getValue() <= 21 && player_hand.getValue() < dealer_hand.getValue())
{
    message = "You loose! (+ player_hand.getValue() + " - " + dealer_hand.getValue() + ")";
    loses++;
    draw.text(300, 310, message);
    ...
}
else if(player_hand.getValue() == dealer_hand.getValue())
{
    message = "You draw! (+ player_hand.getValue() + ")";
    draw.text(300, 310, message);
    ...
}
else
{
    message = "You win! (+ player_hand.getValue() + " - " + dealer_hand.getValue() + ")";
    wins++;
    draw.text(300, 310, message);
    ...
}

In conclusion, I believe that the creation of this game was a success. However, I still have some glitches in the game. First is dealing with Ace and being able to decrement when necessary. Second is that I used 100 pixels between the cards so if the player needs to add more than 4 cards to their hand then the cards would draw off screen. This can easily be remedied, but I did a large padding because it was rare to need to add more than 4 cards. Overall, I enjoyed creating this game and I really learned how to create objects and use different methods on the objects.